Natural Language Processing for Sentiment Analysis

Objective

Build a machine learning model to classify the sentiment of movie reviews (positive/negative).

Data Source

We will use the **IMDb movie review dataset** available through `nltk` for simplicity.

```
import nltk
nltk.download('movie_reviews')
from nltk.corpus import movie_reviews
import random
import numpy as np
import pandas as pd

documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)
len(documents)
```

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
2000
```

Text Preprocessing

Tokenization, stopword removal, and feature extraction using `CountVectorizer`.

```
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


nltk.download('stopwords')
stop_words = stopwords.words('english')

texts = [" ".join(words) for words, label in documents]
labels = [label for words, label in documents]

vectorizer = CountVectorizer(stop_words=stop_words)
X = vectorizer.fit_transform(texts)
y = np.array(labels)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Model Training (Naive Bayes)

```
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
y_pred = model.predict(X_test)
```

Evaluation

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification report : \n",classification_report(y_test, y_pred))
```

```
Accuracy: 0.8325
Confusion Matrix:
 [[180  34]
 [ 33 153]]
Classification report :
               precision    recall  f1-score   support

         neg       0.85      0.84      0.84       214
         pos       0.82      0.82      0.82       186

    accuracy                           0.83       400
   macro avg       0.83      0.83      0.83       400
weighted avg       0.83      0.83      0.83       400
```

Try a Sample Prediction

```
sample = ["This movie was amazing, touching, and beautifully shot"]
sample_vector = vectorizer.transform(sample)
print("Predicted sentiment:", model.predict(sample_vector)[0])
```

```
Predicted sentiment: pos
```

Conclusion

- We successfully built a sentiment classifier using Naive Bayes.
- You can enhance this project by:
  - Using TF-IDF Vectorizer
  - Trying other models like Logistic Regression, SVM, or LSTM
  - Using external datasets (e.g., Twitter sentiment, Amazon reviews)